

16 - 2 G - 00

A

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventorship Tabbara et al.
 Applicant Microsoft Corporation
 Attorney's Docket No. MS1-548US
 Title: System and Method for Restricting Data Transfers and Managing Software Components of
 Distributed Computers

TRANSMITTAL LETTER AND CERTIFICATE OF MAILING

To: Commissioner of Patents and Trademarks,
 Washington, D.C. 20231

From: Allan T. Sponseller (Tel. 509-324-9256; Fax 509-323-8979)
 Lee & Hayes, PLLC
 421 W. Riverside Avenue, Suite 500
 Spokane, WA 99201

The following enumerated items accompany this transmittal letter and are being submitted for the matter identified in the above caption.

1. Specification—title page, plus 50 pages, including 52 claims and Abstract
2. Transmittal letter including Certificate of Express Mailing
3. 9 Sheets Formal Drawings (Figs. 1-9)
4. Return Post Card

Large Entity Status ☒ [x]

Small Entity Status ☐ []

Date: 10-24-2000

By: 

Brian G. Hart
 Reg. No. 44,421

CERTIFICATE OF MAILING

I hereby certify that the items listed above as enclosed are being deposited with the U.S. Postal Service as either first class mail, or Express Mail if the blank for Express Mail No. is completed below, in an envelope addressed to The Commissioner of Patents and Trademarks, Washington, D.C. 20231, on the below-indicated date. Any Express Mail No. has also been marked on the listed items.

Express Mail No. (if applicable) EL685271229

Date: 10/24/00

By: 

Lori A. Vierra

10/24/00
 10662 U.S. PTO

JC926 U.S. PTO
 09/695820
 10/24/00

00505020-102400

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**System and Method for Restricting Data Transfers and
Managing Software Components of Distributed
Computers**

Inventor(s):

**Bassam Tabbara
Galen C. Hunt
Aamer Hydrie
Steven P. Levi
Jakob Rehof
David S. Stutz
Mark D. VanAntwerp
Robert V. Welland**

ATTORNEY'S DOCKET NO. MS1-548US

004207: 02050560

1 **TECHNICAL FIELD**

2 This invention relates to computer system management. More particularly,
3 the invention relates to restricting data transfers and managing software
4 components of distributed computers.

5
6 **BACKGROUND OF THE INVENTION**

7 The Internet and its use have expanded greatly in recent years, and this
8 expansion is expected to continue. One significant way in which the Internet is
9 used is the World Wide Web (also referred to as the "web"), which is a collection
10 of documents (referred to as "web pages") that users can view or otherwise render
11 and which typically include links to one or more other pages that the user can
12 access. Many businesses and individuals have created a presence on the web,
13 typically consisting of one or more web pages describing themselves, describing
14 their products or services, identifying other information of interest, allowing goods
15 or services to be purchased, etc.

16 Web pages are typically made available on the web via one or more web
17 servers, a process referred to as "hosting" the web pages. Sometimes these web
18 pages are freely available to anyone that requests to view them (e.g., a company's
19 advertisements) and other times access to the web pages is restricted (e.g., a
20 password may be necessary to access the web pages). Given the large number of
21 people that may be requesting to view the web pages (especially in light of the
22 global accessibility to the web), a large number of servers may be necessary to
23 adequately host the web pages (e.g., the same web page can be hosted on multiple
24 servers to increase the number of people that can access the web page
25 concurrently). Additionally, because the web is geographically distributed and has

1 non-uniformity of access, it is often desirable to distribute servers to diverse
2 remote locations in order to minimize access times for people in diverse locations
3 of the world. Furthermore, people tend to view web pages around the clock
4 (again, especially in light of the global accessibility to the web), so servers hosting
5 web pages should be kept functional 24 hours per day.

6 Managing a large number of servers, however, can be difficult. A reliable
7 power supply is necessary to ensure the servers can run. Physical security is
8 necessary to ensure that a thief or other mischievous person does not attempt to
9 damage or steal the servers. A reliable Internet connection is required to ensure
10 that the access requests will reach the servers. A proper operating environment
11 (e.g., temperature, humidity, etc.) is required to ensure that the servers operate
12 properly. Thus, "co-location facilities" have evolved which assist companies in
13 handling these difficulties.

14 A co-location facility refers to a complex that can house multiple servers.
15 The co-location facility typically provides a reliable Internet connection, a reliable
16 power supply, and proper operating environment. The co-location facility also
17 typically includes multiple secure areas (e.g., cages) into which different
18 companies can situate their servers. The collection of servers that a particular
19 company situates at the co-location facility is referred to as a "server cluster", even
20 though in fact there may only be a single server at any individual co-location
21 facility. The particular company is then responsible for managing the operation of
22 the servers in their server cluster.

23 Such co-location facilities, however, also present problems. One problem
24 is data security. Different companies (even competitors) can have server clusters
25 at the same co-location facility. Care is required, in such circumstances, to ensure

1 that data received from the Internet (or sent by a server in the server cluster) that is
2 intended for one company is not routed to a server of another company situated at
3 the co-location facility.

4 An additional problem is the management of the servers once they are
5 placed in the co-location facility. Currently, a system administrator from a
6 company is able to contact a co-location facility administrator (typically by
7 telephone) and ask him or her to reset a particular server (typically by pressing a
8 hardware reset button on the server, or powering off then powering on the server)
9 in the event of a failure of (or other problem with) the server. This limited reset-
10 only ability provides very little management functionality to the company.
11 Alternatively, the system administrator from the company can physically travel to
12 the co-location facility him/her-self and attend to the faulty server. Unfortunately,
13 a significant amount of time can be wasted by the system administrator in
14 traveling to the co-location facility to attend to a server. Thus, it would be
15 beneficial to have an improved way to manage server computers at a co-location
16 facility.

17 Additionally, the world is becoming populated with ever increasing
18 numbers of individual user computers in the form of personal computers (PCs),
19 personal digital assistants (PDAs), pocket computers, palm-sized computers,
20 handheld computers, digital cellular phones, etc. Management of the software on
21 these user computers can be very laborious and time consuming and is particularly
22 difficult for the often non-technical users of these machines. Often a system
23 administrator or technician must either travel to the remote location of the user's
24 computer, or walk through management operations over a telephone. It would be
25

1 further beneficial to have an improved way to manage remote computers at the
2 user's location without user intervention.

3 The invention described below addresses these disadvantages, restricting
4 data transfers and managing software components of distributed computers.

5 6 **SUMMARY OF THE INVENTION**

7 Restricting data transfers and managing software components in clusters of
8 server computers located at a co-location facility is described herein.

9 According to one aspect, a controller (referred to as the "BMonitor") is
10 situated on a computer (e.g., each node in a co-location facility). The BMonitor
11 includes a plurality of filters that identify where data can be sent to and/or received
12 from, such as another node in the co-location facility or a client computer coupled
13 to the computer via the Internet. These filters can then be modified, during
14 operation of the computer, by one or more management devices coupled to the
15 computer.

16 According to another aspect, a controller referred to as the "BMonitor"
17 (situated on a computer) manages software components executing on that
18 computer. Requests are received by the BMonitor from external sources and
19 implemented by the BMonitor. Such requests can originate from a management
20 console local to the computer or alternatively remote from the computer.

21 According to another aspect, a controller referred to as the "BMonitor"
22 (situated on a computer) operates as a trusted third party mediating interaction
23 among multiple management devices. The BMonitor maintains multiple
24 ownership domains, each corresponding to a management device(s) and each
25 having a particular set of rights that identify what types of management functions

they can command the BMonitor to carry out. Only one ownership domain is the top-level domain at any particular time, and the top-level domain has a more expanded set of rights than any of the lower-level domains. The top-level domain can create new ownership domains corresponding to other management device, and can also be removed and the management rights of its corresponding management device revoked at any time by a management device corresponding to a lower-level ownership domain. Each time a change of which ownership domain is the top-level ownership domain occurs, the computer's system memory can be erased so that no confidential information from one ownership domain is made available to devices corresponding to other ownership domains.

According to another aspect, the BMonitor is implemented in a more-privileged level than other software engines executing on the node, preventing other software engines from interfering with restrictions imposed by the BMonitor.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings. The same numbers are used throughout the figures to reference like components and/or features.

Fig. 1 shows a client/server network system and environment such as may be used with certain embodiments of the invention.

Fig. 2 shows a general example of a computer that can be used in accordance with certain embodiments of the invention.

Fig. 3 is a block diagram illustrating an exemplary co-location facility in more detail.

1 Fig. 4 is a block diagram illustrating an exemplary multi-tiered server
2 cluster management architecture.

3 Fig. 5 is a block diagram illustrating an exemplary node of a co-location
4 facility in more detail in accordance with certain embodiments of the invention.

5 Fig. 6 is a block diagram illustrating an exemplary set of ownership
6 domains in accordance with certain embodiments of the invention.

7 Fig. 7 is a flow diagram illustrating the general operation of a BMonitor in
8 accordance with certain embodiments of the invention.

9 Fig. 8 is a flowchart illustrating an exemplary process for handling
10 outbound data requests in accordance with certain embodiments of the invention.

11 Fig. 9 is a flowchart illustrating an exemplary process for handling inbound
12 data requests in accordance with certain embodiments of the invention.

13 14 **DETAILED DESCRIPTION**

15 Fig. 1 shows a client/server network system and environment such as may
16 be used with certain embodiments of the invention. Generally, the system
17 includes one or more (n) client computers 102, one or more (m) co-location
18 facilities 104 each including multiple clusters of server computers (server clusters)
19 106, one or more management devices 110, and one or more separate (e.g., not
20 included in a co-location facility) servers 112. The servers, clients, and
21 management devices communicate with each other over a data communications
22 network 108. The communications network in Fig. 1 comprises a public network
23 108 such as the Internet. Other types of communications networks might also be
24 used, in addition to or in place of the Internet, including local area networks
25 (LANs), wide area networks (WANs), etc. Data communications network 108 can

1 be implemented in any of a variety of different manners, including wired and/or
2 wireless communications media.

3 Communication over network 108 can be carried out using any of a wide
4 variety of communications protocols. In one implementation, client computers
5 102 and server computers in clusters 106 can communicate with one another using
6 the Hypertext Transfer Protocol (HTTP), in which web pages are hosted by the
7 server computers and written in a markup language, such as the Hypertext Markup
8 Language (HTML) or the eXtensible Markup Language (XML).

9 Management device 110 operates to manage software components of one or
10 more computing devices located at a location remote from device 110. This
11 management may also include restricting data transfers into and/or out of the
12 computing device being managed. In the illustrated example of Fig. 1,
13 management device 110 can remotely manage any one or more of: a client(s) 102,
14 a server cluster(s) 106, or a server(s) 112. Any of a wide variety of computing
15 devices can be remotely managed, including personal computers (PCs), network
16 PCs, multiprocessor systems, microprocessor-based or programmable consumer
17 electronics, minicomputers, mainframe computers, gaming consoles, Internet
18 appliances, personal digital assistants (PDAs), pocket computers, palm-sized
19 computers, handheld computers, digital cellular phones, etc. Remote management
20 of a computing device is accomplished by communicating commands to the
21 device via network 108, as discussed in more detail below.

22 In the discussion herein, embodiments of the invention are described in the
23 general context of computer-executable instructions, such as program modules,
24 being executed by one or more conventional personal computers. Generally,
25 program modules include routines, programs, objects, components, data structures,

0012345678910111213141516171819202122232425
etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that various embodiments of the invention may be practiced with other computer system configurations, including hand-held devices, gaming consoles, Internet appliances, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. In a distributed computer environment, program modules may be located in both local and remote memory storage devices.

Alternatively, embodiments of the invention can be implemented in hardware or a combination of hardware, software, and/or firmware. For example, all or part of the invention can be implemented in one or more application specific integrated circuits (ASICs) or programmable logic devices (PLDs).

Fig. 2 shows a general example of a computer 142 that can be used in accordance with certain embodiments of the invention. Computer 142 is shown as an example of a computer that can perform the functions of a client computer 102 of Fig. 1, a server computer or node in a co-location facility 104 of Fig. 1, a management device 110 of Fig. 1, a server 112 of Fig. 1, or a local or remote management console as discussed in more detail below.

Computer 142 includes one or more processors or processing units 144, a system memory 146, and a bus 148 that couples various system components including the system memory 146 to processors 144. The bus 148 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. The system memory includes read only memory (ROM) 150 and random access memory (RAM) 152.

1 A basic input/output system (BIOS) 154, containing the basic routines that help to
2 transfer information between elements within computer 142, such as *during start-*
3 *up*, is stored in ROM 150.

4 Computer 142 further includes a hard disk drive 156 for reading from and
5 writing to a hard disk, not shown, connected to bus 148 via a hard disk driver
6 interface 157 (e.g., a SCSI, ATA, or other type of interface); a magnetic disk drive
7 158 for reading from and writing to a removable magnetic disk 160, connected to
8 bus 148 via a magnetic disk drive interface 161; and an optical disk drive 162 for
9 reading from or writing to a removable optical disk 164 such as a CD ROM, DVD,
10 or other optical media, connected to bus 148 via an optical drive interface 165.
11 The drives and their associated computer-readable media provide nonvolatile
12 storage of computer readable instructions, data structures, program modules and
13 other data for computer 142. Although the exemplary environment described
14 herein employs a hard disk, a removable magnetic disk 160 and a removable
15 optical disk 164, it should be appreciated by those skilled in the art that other types
16 of computer readable media which can store data that is accessible by a computer,
17 such as magnetic cassettes, flash memory cards, digital video disks, random access
18 memories (RAMs) read only memories (ROM), and the like, may also be used in
19 the exemplary operating environment.

20 A number of program modules may be stored on the hard disk, magnetic
21 disk 160, optical disk 164, ROM 150, or RAM 152, including an operating system
22 170, one or more application programs 172, other program modules 174, and
23 program data 176. A user may enter commands and information into computer
24 142 through input devices such as keyboard 178 and pointing device 180. Other
25 input devices (not shown) may include a microphone, joystick, game pad, satellite

1 dish, scanner, or the like. These and other input devices are connected to the
2 processing unit 144 through an interface 168 that is coupled to the system bus. A
3 monitor 184 or other type of display device is also connected to the system bus
4 148 via an interface, such as a video adapter 186. In addition to the monitor,
5 personal computers typically include other peripheral output devices (not shown)
6 such as speakers and printers.

7 Computer 142 optionally operates in a networked environment using
8 logical connections to one or more remote computers, such as a remote computer
9 188. The remote computer 188 may be another personal computer, a server, a
10 router, a network PC, a peer device or other common network node, and typically
11 includes many or all of the elements described above relative to computer 142,
12 although only a memory storage device 190 has been illustrated in Fig. 2. The
13 logical connections depicted in Fig. 2 include a local area network (LAN) 192 and
14 a wide area network (WAN) 194. Such networking environments are
15 commonplace in offices, enterprise-wide computer networks, intranets, and the
16 Internet. In the described embodiment of the invention, remote computer 188
17 executes an Internet Web browser program (which may optionally be integrated
18 into the operating system 170) such as the "Internet Explorer" Web browser
19 manufactured and distributed by Microsoft Corporation of Redmond, Washington.

20 When used in a LAN networking environment, computer 142 is connected
21 to the local network 192 through a network interface or adapter 196. When used
22 in a WAN networking environment, computer 142 typically includes a modem 198
23 or other component for establishing communications over the wide area network
24 194, such as the Internet. The modem 198, which may be internal or external, is
25 connected to the system bus 148 via an interface (e.g., a serial port interface 168).

1 In a networked environment, program modules depicted relative to the personal
2 computer 142, or portions thereof, may be stored in the remote memory storage
3 device. It is to be appreciated that the network connections shown are exemplary
4 and other means of establishing a communications link between the computers
5 may be used.

6 Generally, the data processors of computer 142 are programmed by means
7 of instructions stored at different times in the various computer-readable storage
8 media of the computer. Programs and operating systems are typically distributed,
9 for example, on floppy disks or CD-ROMs. From there, they are installed or
10 loaded into the secondary memory of a computer. At execution, they are loaded at
11 least partially into the computer's primary electronic memory. The invention
12 described herein includes these and other various types of computer-readable
13 storage media when such media contain instructions or programs for implementing
14 the steps described below in conjunction with a microprocessor or other data
15 processor. The invention also includes the computer itself when programmed
16 according to the methods and techniques described below. Furthermore, certain
17 sub-components of the computer may be programmed to perform the functions
18 and steps described below. The invention includes such sub-components when
19 they are programmed as described. In addition, the invention described herein
20 includes data structures, described below, as embodied on various types of
21 memory media.

22 For purposes of illustration, programs and other executable program
23 components such as the operating system are illustrated herein as discrete blocks,
24 although it is recognized that such programs and components reside at various
25

1 times in different storage components of the computer, and are executed by the
2 data processor(s) of the computer.

3 Fig. 3 is a block diagram illustrating an exemplary co-location facility in
4 more detail. Co-location facility 104 is illustrated including multiple nodes (also
5 referred to as server computers) 210. Co-location facility 104 can include any
6 number of nodes 210, and can easily include an amount of nodes numbering into
7 the thousands.

8 The nodes 210 are grouped together in clusters, referred to as server
9 clusters (or node clusters). For ease of explanation and to avoid cluttering the
10 drawings, only a single cluster 212 is illustrated in Fig. 3. Each server cluster
11 includes nodes 210 that correspond to a particular customer of co-location facility
12 104. The nodes 210 of a server cluster can be physically isolated from the nodes
13 210 of other server clusters. This physical isolation can take different forms, such
14 as separate locked cages or separate rooms at co-location facility 104. Physically
15 isolating server clusters ensures customers of co-location facility 104 that only
16 they can physically access their nodes (other customers cannot).

17 A landlord/tenant relationship (also referred to as a lessor/lessee
18 relationship) can also be established based on the nodes 210. The owner (and/or
19 operator) of co-location facility 104 owns (or otherwise has rights to) the
20 individual nodes 210, and thus can be viewed as a "landlord". The customers of
21 co-location facility 104 lease the nodes 210 from the landlord, and thus can be
22 viewed as a "tenant". The landlord is typically not concerned with what types of
23 data or programs are being stored at the nodes 210 by the tenant, but does impose
24 boundaries on the clusters that prevent nodes 210 from different clusters from
25 communicating with one another, as discussed in more detail below. Additionally,

1 the nodes 210 provide assurances to the tenant that, although the nodes are only
2 leased to the tenant, the landlord cannot access confidential information stored by
3 the tenant.

4 Although physically isolated, nodes 210 of different clusters are often
5 physically coupled to the same transport medium (or media) 211 that enables
6 access to network connection(s) 216, and possibly application operations
7 management console 242, discussed in more detail below. This transport medium
8 can be wired or wireless.

9 As each node 210 can be coupled to a shared transport medium 211, each
10 node 210 is configurable to restrict which other nodes 210 data can be sent to or
11 received from. Given that a number of different nodes 210 may be included in a
12 customer's (also referred to as tenant's) server cluster, the customer may want to be
13 able to pass data between different nodes 210 within the cluster for processing,
14 storage, etc. However, the customer will typically not want data to be passed to
15 other nodes 210 that are not in the server cluster. Configuring each node 210 in
16 the cluster to restrict which other nodes 210 data can be sent to or received from
17 allows a boundary for the server cluster to be established and enforced.
18 Establishment and enforcement of such server cluster boundaries prevents
19 customer data from being erroneously or improperly forwarded to a node that is
20 not part of the cluster.

21 These initial boundaries established by the landlord prevent communication
22 between nodes 210 of different customers, thereby ensuring that each customer's
23 data can be passed to other nodes 210 of that customer. The customer itself may
24 also further define sub-boundaries within its cluster, establishing sub-clusters of
25 nodes 210 that data cannot be communicated out of (or in to) either to or from

1 other nodes in the cluster. The customer is able to add, modify, remove, etc. such
2 sub-cluster boundaries at will, but only within the boundaries defined by the
3 landlord (that is, the cluster boundaries). Thus, the customer is not able to alter
4 boundaries in a manner that would allow communication to or from a node 210 to
5 extend to another node 210 that is not within the same cluster.

6 Co-location facility 104 supplies reliable power 214 and reliable network
7 connection(s) 216 (e.g., to network 108 of Fig. 1) to each of the nodes 210. Power
8 214 and network connection(s) 216 are shared by all of the nodes 210, although
9 alternatively separate power 214 and network connection(s) 216 may be supplied
10 to nodes 210 or groupings (e.g., clusters) of nodes. Any of a wide variety of
11 conventional mechanisms for supplying reliable power can be used to supply
12 reliable power 214, such as power received from a public utility company along
13 with backup generators in the event of power failures, redundant generators,
14 batteries, fuel cells, or other power storage mechanisms, etc. Similarly, any of a
15 wide variety of conventional mechanisms for supplying a reliable network
16 connection can be used to supply network connection(s) 216, such as redundant
17 connection transport media, different types of connection media, different access
18 points (e.g., different Internet access points, different Internet service providers
19 (ISPs), etc.).

20 In certain embodiments, nodes 210 are leased or sold to customers by the
21 operator or owner of co-location facility 104 along with the space (e.g., locked
22 cages) and service (e.g., access to reliable power 214 and network connection(s)
23 216) at facility 104. In other embodiments, space and service at facility 104 may
24 be leased to customers while one or more nodes are supplied by the customer.
25

Management of each node 210 is carried out in a multiple-tiered manner. Fig. 4 is a block diagram illustrating an exemplary multi-tiered management architecture. The multi-tiered architecture includes three tiers: a cluster operations management tier 230, an application operations management tier 232, and an application development tier 234. Cluster operations management tier 230 is implemented locally at the same location as the server(s) being managed (e.g., at a co-location facility) and involves managing the hardware operations of the server(s). In the illustrated example, cluster operations management tier 230 is not concerned with what software components are executing on the nodes 210, but only with the continuing operation of the hardware of nodes 210 and establishing any boundaries between clusters of nodes.

The application operations management tier 232, on the other hand, is implemented at a remote location other than where the server(s) being managed are located (e.g., other than the co-location facility), but from a client computer that is still communicatively coupled to the server(s). The application operations management tier 232 involves managing the software operations of the server(s) and defining any sub-boundaries within server clusters. The client can be coupled to the server(s) in any of a variety of manners, such as via the Internet or via a dedicated (e.g., dial-up) connection. The client can be coupled continually to the server(s), or alternatively sporadically (e.g., only when needed for management purposes).

The application development tier 234 is implemented on another client computer at a location other than the server(s) (e.g., other than at the co-location facility) and involves development of software components or engines for execution on the server(s). Alternatively, current software on a node 210 at co-

1 operations management console may correspond to, and manage hardware
2 operations of, multiple server clusters. According to another alternative, multiple
3 cluster operations management consoles may correspond to, and manage hardware
4 operations of, a single server cluster. Such multiple consoles can manage a single
5 server cluster in a shared manner, or one console may operate as a backup for
6 another console (e.g., providing increased reliability through redundancy, to allow
7 for maintenance, etc.).

8 An application operations management console 242 is also
9 communicatively coupled to co-location facility 104. Application operations
10 management console 242 may be, for example, a management device 110 of Fig.
11 1. Application operations management console 242 is located at a location remote
12 from co-location facility 104 (that is, not within co-location facility 104), typically
13 being located at the offices of the customer. A different application operations
14 management console 242 corresponds to each server cluster of co-location facility
15 104, although alternatively multiple consoles 242 may correspond to a single
16 server cluster, or a single console 242 may correspond to multiple server clusters.
17 Application operations management console 240 implements application
18 operations management tier 232 (Fig. 4) for cluster 212 and is responsible for
19 managing the software operations of nodes 210 in cluster 212 as well as securing
20 sub-boundaries within cluster 212.

21 Application operations management console 242 monitors the software in
22 cluster 212 and attempts to identify software failures. Any of a wide variety of
23 software failures can be monitored for, such as application processes or threads
24 that are "hung" or otherwise non-responsive, an error in execution of application
25 processes or threads, etc. Software operations can be monitored in any of a variety

1 of manners (similar to the monitoring of hardware operations discussed above),
2 such as application operations management console 242 sending test messages or
3 control signals to particular processes or threads executing on the nodes 210 that
4 require the use of particular routines in order to respond (no response or an
5 incorrect response indicates failure), having messages or control signals that
6 require the use of particular software routines to generate periodically sent by
7 processes or threads executing on nodes 210 to application operations
8 management console 242 (not receiving such a message or control signal within a
9 specified amount of time indicates failure), etc. Alternatively, application
10 operations management console 242 may make no attempt to identify what type of
11 software failure has occurred, but rather simply that a failure has occurred.

12 Once a software failure is detected, application operations management
13 console 242 acts to correct the failure. The action taken by application operations
14 management console 242 can vary based on the hardware as well as the type of
15 failure, and can vary for different server clusters. The corrective action can be
16 notification of an administrator (e.g., a flashing light, an audio alarm, an electronic
17 mail message, calling a cell phone or pager, etc.), or an attempt to correct the
18 problem (e.g., reboot the node, re-load the software component or engine image,
19 terminate and re-execute the process, etc.).

20 Thus, the management of a node 210 is distributed across multiple
21 managers, regardless of the number of other nodes (if any) situated at the same
22 location as the node 210. The multi-tiered management allows the hardware
23 operations management to be separated from the application operations
24 management, allowing two different consoles (each under the control of a different
25 entity) to share the management responsibility for the node.

Fig. 5 is a block diagram illustrating an exemplary remotely managed node in more detail in accordance with certain embodiments of the invention. Node 248 can be a node 210 of a co-location facility, or alternatively a separate device (e.g., a client 102 or server 112 of Fig. 1). Node 248 includes a monitor 250, referred to as the "BMonitor", and a plurality of software components or engines 252, and is coupled to (or alternatively incorporates) a mass storage device 262. In the illustrated example, node 248 is a computing device having a processor(s) that supports multiple privilege levels (e.g., rings in an x86 architecture processor). In the illustrated example, these privilege levels are referred to as rings, although alternate implementations using different processor architectures may use different nomenclature. The multiple rings provide a set of prioritized levels that software can execute at, often including 4 levels (Rings 0, 1, 2, and 3). Ring 0 is typically referred to as the most privileged ring. Software processes executing in Ring 0 can typically access more features (e.g., instructions) than processes executing in less privileged Rings. Furthermore, a processor executing in a particular Ring cannot alter code or data in a higher priority ring. In the illustrated example, BMonitor 250 executes in Ring 0, while engines 252 execute in Ring 1 (or alternatively Rings 2 and/or 3). Thus, the code or data of BMonitor 250 (executing in Ring 0) cannot be altered directly by engines 252 (executing in Ring 1). Rather, any such alterations would have to be made by an engine 252 requesting BMonitor 250 to make the alteration (e.g., by sending a message to BMonitor 250, invoking a function of BMonitor 250, etc.). Implementing BMonitor 250 in Ring 0 protects BMonitor 250 from a rogue or malicious engine 252 that tries to bypass any restrictions imposed by BMonitor 250.

Alternatively, BMonitor 250 may be implemented in other manners that protect it from a rogue or malicious engine 252. For example, node 248 may include multiple processors – one (or more) processor(s) for executing engines 252, and another processor(s) to execute BMonitor 250. By allowing only BMonitor 250 to execute on a processor(s) separate from the processor(s) on which engines 252 are executing, BMonitor 250 can be effectively shielded from engines 252.

BMonitor 250 is the fundamental control module of node 248 – it controls (and optionally includes) both the network interface card and the memory manager. By controlling the network interface card (which may be separate from BMonitor 250, or alternatively BMonitor 250 may be incorporated on the network interface card), BMonitor 250 can control data received by and sent by node 248. By controlling the memory manager, BMonitor 250 controls the allocation of memory to engines 252 executing in node 248 and thus can assist in preventing rogue or malicious engines from interfering with the operation of BMonitor 250.

Although various aspects of node 248 may be under control of BMonitor 250 (e.g., the network interface card), BMonitor 250 still makes at least part of such functionality available to engines 252 executing on the node 248. BMonitor 250 provides an interface (e.g., via controller 254 discussed in more detail below) via which engines 252 can request access to the functionality, such as to send data out to another node 248 within a co-location facility or on the Internet. These requests can take any of a variety of forms, such as sending messages, calling a function, etc.

BMonitor 250 includes controller 254, network interface 256, one or more filters 258, one or more keys 259, and a BMonitor Control Protocol (BMCP)

1 module 260. Network interface 256 provides the interface between node 248 and
2 the network (e.g., network 108 of Fig. 1). Filters 258 identify other nodes 248 in a
3 co-location facility (and/or other sources or targets (e.g., coupled to Internet 108 of
4 Fig. 1) that data can (or alternatively cannot) be sent to and/or received from. The
5 nodes or other sources/targets can be identified in any of a wide variety of
6 manners, such as by network address (e.g., Internet Protocol (IP) address), some
7 other globally unique identifier, a locally unique identifier (e.g., a numbering
8 scheme proprietary or local to co-location facility 104), etc.

9 Filters 258 can fully restrict access to a node (e.g., no data can be received
10 from or sent to the node), or partially restrict access to a node. Partial access
11 restriction can take different forms. For example, a node may be restricted so that
12 data can be received from the node but not sent to the node (or vice versa). By
13 way of another example, a node may be restricted so that only certain types of data
14 (e.g., communications in accordance with certain protocols, such as HTTP) can be
15 received from and/or sent to the node. Filtering based on particular types of data
16 can be implemented in different manners, such as by communicating data in
17 packets with header information that indicate the type of data included in the
18 packet.

19 Filters 258 can be added by one or more management devices 110 of Fig. 1
20 or either of application operations management console 242 or cluster operations
21 management console 240 of Fig. 3. In the illustrated example, filters added by
22 cluster operations management console 240 (to establish cluster boundaries)
23 restrict full access to nodes (e.g., any access to another node can be prevented)
24 whereas filters added by application operations management console 242 (to
25

1 establish sub-boundaries within a cluster) or management device 110 can restrict
2 either full access to nodes or partial access.

3 Controller 254 also imposes some restrictions on what filters can be added
4 to filters 258. In the multi-tiered management architecture illustrated in Figs. 3
5 and 4, controller 254 allows cluster operations management console 240 to add
6 any filters it desires (which will define the boundaries of the cluster). However,
7 controller 254 restricts application operations management console 242 to adding
8 only filters that are at least as restrictive as those added by console 240. If console
9 242 attempts to add a filter that is less restrictive than those added by console 240
10 (in which case the sub-boundary may extend beyond the cluster boundaries),
11 controller 254 refuses to add the filter (or alternatively may modify the filter so
12 that it is not less restrictive). By imposing such a restriction, controller 254 can
13 ensure that the sub-boundaries established at the application operations
14 management level do not extend beyond the cluster boundaries established at the
15 cluster operations management level.

16 Controller 254, using one or more filters 258, operates to restrict data
17 packets sent from node 248 and/or received by node 248. All data intended for an
18 engine 252, or sent by an engine 252, to another node, is passed through network
19 interface 256 and filters 258. Controller 254 applies the filters 258 to the data,
20 comparing the target of the data (e.g., typically identified in a header portion of a
21 packet including the data) to acceptable (and/or restricted) nodes (and/or network
22 addresses) identified in filters 258. If filters 258 indicate that the target of the data
23 is acceptable, then controller 254 allows the data to pass through to the target
24 (either into node 248 or out from node 248). However, if filters 258 indicate that
25 the target of the data is not acceptable, then controller 254 prevents the data from

1 passing through to the target. Controller 254 may return an indication to the
2 source of the data that the data cannot be passed to the target, or may simply
3 ignore or discard the data.

4 The application of filters 258 to the data by controller 254 allows the
5 boundary restrictions of a server cluster (Fig. 3) to be imposed. Filters 258 can be
6 programmed (e.g., by application operations management console 242 of Fig. 3)
7 with the node addresses of all the nodes within the server cluster (e.g., cluster
8 212). Controller 254 then prevents data received from any node not within the
9 server cluster from being passed through to an engine 252, and similarly prevents
10 any data being sent to a node other than one within the server cluster from being
11 sent. Similarly, data received from Internet 108 (Fig. 1) can identify a target node
12 248 (e.g., by IP address), so that controller 254 of any node other than the target
13 node will prevent the data from being passed through to an engine 252.
14 Furthermore, as filters 258 can be readily modified by cluster operations
15 management console 240, server cluster boundaries can be easily changed to
16 accommodate changes in the server cluster (e.g., addition of nodes to and/or
17 removal of nodes from the server cluster).

18 BMCP module 260 implements the Distributed Host Control Protocol
19 (DHCP), allowing BMonitor 250 (and thus node 248) to obtain an IP address from
20 a DHCP server (e.g., cluster operations management console 240 of Fig. 3).
21 During an initialization process for node 248, BMCP module 260 requests an IP
22 address from the DHCP server, which in turn provides the IP address to module
23 260. Additional information regarding DHCP is available from Microsoft
24 Corporation of Redmond, Washington.

Software engines 252 include any of a wide variety of conventional software components. Examples of engines 252 include an operating system (e.g., Windows NT®), a load balancing server component (e.g., to balance the processing load of multiple nodes 248), a caching server component (e.g., to cache data and/or instructions from another node 248 or received via the Internet), a storage manager component (e.g., to manage storage of data from another node 248 or received via the Internet), etc. In one implementation, each of the engines 252 is a protocol-based engine, communicating with BMonitor 250 and other engines 252 via messages and/or function calls without requiring the engines 252 and BMonitor 250 to be written using the same programming language.

Controller 254, in conjunction with loader 264, is responsible for controlling the execution of engines 252. This control can take different forms, including beginning or initiating execution of an engine 252, terminating execution of an engine 252, re-loading an image of an engine 252 from a storage device, debugging execution of an engine 252, etc. Controller 254 receives instructions from application operations management console 242 of Fig. 3 or a management device(s) 110 of Fig. 1 regarding which of these control actions to take and when to take them. In the event that execution of an engine 252 is to be initiated (including re-starting an engine whose execution was recently terminated), controller 254 communicates with loader 264 to load an image of the engine 252 from a storage device (e.g., device 262, ROM, etc.) into the memory (e.g., RAM) of node 248. Loader 264 operates in a conventional manner to copy the image of the engine from the storage device into memory and initialize any necessary operating system parameters to allow execution of the engine 252.

1 Thus, the control of engines 252 is actually managed by a remote device, not
2 locally at the same location as the node 248 being managed.

3 Controller 254 also provides an interface via which application operations
4 management console 242 of Fig. 3 or a management device(s) 110 of Fig. 1 can
5 identify filters to add (and/or remove) from filter set 258.

6 Controller 254 also includes an interface via which cluster operations
7 management console 240 of Fig. 3 can communicate commands to controller 254.
8 Different types of hardware operation oriented commands can be communicated to
9 controller 254 by cluster operations management console 240, such as re-booting
10 the node, shutting down the node, placing the node in a low-power state (e.g., in a
11 suspend or standby state), changing cluster boundaries, changing encryption keys
12 (if any), etc.

13 Controller 254 further optionally provides encryption support for BMonitor
14 250, allowing data to be stored securely on mass storage device 262 (e.g., a
15 magnetic disk, an optical disk, etc.) and secure communications to occur between
16 node 248 and an operations management console (e.g., console 240 or 242 of Fig.
17 3) or other management device (e.g., management device 110 of Fig. 1).
18 Controller 254 maintains multiple encryption keys 259, which can include a
19 variety of different keys such as symmetric keys (secret keys used in secret key
20 cryptography), public/private key pairs (for public key cryptography), etc. to be
21 used in encrypting and/or decrypting data.

22 BMonitor 250 makes use of public key cryptography to provide secure
23 communications between node 248 and the management consoles (e.g., consoles
24 240 or 242 of Fig. 3) or other management devices (e.g., management device(s)
25 110 of Fig. 1). Public key cryptography is based on a key pair, including both a

004207-0256960

1 public key and a private key, and an encryption algorithm. The encryption
2 algorithm can encrypt data based on the public key such that it cannot be
3 decrypted efficiently without the private key. Thus, communications from the
4 public-key holder can be encrypted using the public key, allowing only the
5 private-key holder to decrypt the communications. Any of a variety of public key
6 cryptography techniques may be used, such as the well-known RSA (Rivest,
7 Shamir, and Adelman) encryption technique. For a basic introduction of
8 cryptography, the reader is directed to a text written by Bruce Schneier and
9 entitled "Applied Cryptography: Protocols, Algorithms, and Source Code in C,"
10 published by John Wiley & Sons with copyright 1994 (or second edition with
11 copyright 1996).

12 BMonitor 250 is initialized to include a public/private key pair for both the
13 landlord and the tenant. These key pairs can be generated by BMonitor 250, or
14 alternatively by some other component and stored within BMonitor 250 (with that
15 other component being trusted to destroy its knowledge of the key pair). As used
16 herein, U refers to a public key and R refers to a private key. The public/private
17 key pair for the landlord is referred to as (U_L, R_L) , and the public/private key pair
18 for the tenant is referred to as (U_T, R_T) . BMonitor 250 makes the public keys U_L
19 and U_T available to the landlord, but keeps the private keys R_L and R_T secret. In
20 the illustrated example, BMonitor 250 never divulges the private keys R_L and R_T ,
21 so both the landlord and the tenant can be assured that no entity other than the
22 BMonitor 250 can decrypt information that they encrypt using their public keys
23 (e.g., via cluster operations management console 240 and application operations
24 management console 242 of Fig. 3, respectively).

1 Once the landlord has the public keys U_L and U_T , the landlord can assign
2 node 248 to a particular tenant, giving that tenant the public key U_T . Use of the
3 public key U_T allows the tenant to encrypt communications to BMonitor 250 that
4 only BMonitor 250 can decrypt (using the private key R_T). Although not required,
5 a prudent initial step for the tenant is to request that BMonitor 250 generate a new
6 public/private key pair (U_T , R_T). In response to such a request, controller 254 or a
7 dedicated key generator (not shown) of BMonitor 250 generates a new
8 public/private key pair in any of a variety of well-known manners, stores the new
9 key pair as the tenant key pair, and returns the new public key U_T to the tenant. By
10 generating a new key pair, the tenant is assured that no other entity, including the
11 landlord, is aware of the tenant public key U_T . Additionally, the tenant may also
12 have new key pairs generated at subsequent times.

13 Having a public/private key pair in which BMonitor 250 stores the private
14 key and the tenant knows the public key allows information to be securely
15 communicated from the tenant to BMonitor 250. In order to ensure that
16 information can be securely communicated from BMonitor 250 to the tenant, an
17 additional public/private key pair is generated by the tenant and the public key
18 portion is communicated to BMonitor 250. Any communications from BMonitor
19 250 to the tenant can thus be encrypted using this public key portion, and can be
20 decrypted only by the holder of the corresponding private key (that is, only by the
21 tenant).

22 BMonitor 250 also maintains, as one of keys 259, a disk key which is
23 generated based on one or more symmetric keys (symmetric keys refer to secret
24 keys used in secret key cryptography). The disk key, also a symmetric key, is
25 used by BMonitor 250 to store information in mass storage device 262. BMonitor

(removed from) stack 286 along with any other higher-level ownership domains. For example, if the owner of node 248 (ownership domain 280) were to revoke the rights of ownership domain 282, then ownership domains 282 and 284 would be popped from ownership domain stack 286.

Each ownership domain has a corresponding set of rights. In the illustrated example, the top-level ownership domain has one set of rights that include: (1) the right to push new ownership domains on the ownership domain stack; (2) the right to access any system memory in the node; (3) the right to access any mass storage devices in or coupled to the node; (4) the right to modify (add, remove, or change) packet filters at the node; (5) the right to start execution of software engines on the node (e.g., engines 252 of Fig. 5); (6) the right to stop execution of software engines on the node, including resetting the node; (7) the right to debug software engines on the node; (8) the right to change its own authentication credentials (e.g., its public key or ID); (9) the right to modify its own storage key; (10) the right to subscribe to events engine events, machine events, and/or packet filter events (e.g., notify a management console or other device when one of these events occurs). Additionally, each of the lower-level ownership domains has another set of rights that include: (1) the right to pop an existing ownership domain(s); (2) the right to modify (add, remove, or change) packet filters at the node; (3) the right to change its own authentication credentials (e.g., public key or ID); and (4) the right to subscribe to machine events and/or packet filter events. Alternatively, some of these rights may not be included (e.g., depending on the situation, the right to debug software engines on the node may not be needed), or other rights may be included (e.g., the top-level node may include the right to pop itself off the ownership domain stack).

Ownership domains can be added to and removed from ownership domain stack 286 numerous times during operation. Which ownership domains are removed and/or added varies based on the activities being performed. By way of example, if the owner of node 248 (corresponding to root ownership domain 280) desires to perform some operation on node 248, all higher-level ownership domains 282 – 284 are revoked, the desired operation is performed (ownership domain 280 is now the top-level domain, so the expanded set of rights are available), and then new ownership domains can be created and added to ownership domain stack 286 (e.g., so that the management agent previously corresponding to the top-level ownership domain is returned to its previous position).

BMonitor 250 checks, for each request received from an entity corresponding to one of the ownership domains (e.g., a management console controlled by the entity), what rights the ownership domain has. If the ownership domain has the requisite rights for the request to be implemented, then BMonitor 250 carries out the request. However, if the ownership domain does not have the requisite set of rights, then the request is not carried out (e.g., an indication that the request cannot be carried out can be returned to the requestor, or alternatively the request can simply be ignored).

In the illustrated example, each ownership domain includes an identifier (ID), a public key, and a storage key. The identifier serves as a unique identifier of the ownership domain, the public key is used to send secure communications to a management device corresponding to the ownership domain, and the storage key is used (at least in part) to encrypt information stored on mass storage devices. An additional private key may also be included for each ownership domain for the

1 using its private key, and then returns the decrypted token to BMonitor 250. If the
2 returned token matches the token that BMonitor 250 generated, then the
3 authenticity of the management device is verified (because only the management
4 device with the corresponding private key would be able to decrypt the token). An
5 analogous process can be used for BMonitor 250 to authenticate itself to the
6 management device.

7 Once authenticated, the management device can communicate requests to
8 BMonitor 250 and have any of those requests carried out (assuming it has the
9 rights to do so). Although not required, it is typically prudent for a management
10 console, upon initially authenticating itself to BMonitor 250, to change its public
11 key/private key pair.

12 When a new ownership domain is created, the management device that is
13 creating the new ownership domain can optionally terminate any executing
14 engines 252 and erase any system memory and mass storage devices. This
15 provides an added level of security, on top of the encryption, to ensure that one
16 management device does not have access to information stored on the hardware by
17 another management device. Additionally, each time an ownership domain is
18 popped from the stack, BMonitor 250 terminated any executing engines 252,
19 erases the system memory, and also erases the storage key for that ownership
20 domain. Thus, any information stored by that ownership domain cannot be
21 accessed by the remaining ownership domains – the memory has been erased so
22 there is no data in memory, and without the storage key information on the mass
23 storage device cannot be decrypted. BMonitor 250 may alternatively erase the
24 mass storage device too. However, by simply erasing the key and leaving the data
25

1 encrypted, BMonitor 250 allows the data to be recovered if the popped ownership
2 domain is re-created (and uses the same storage key).

3 Fig. 7 is a flow diagram illustrating the general operation of BMonitor 250
4 in accordance with certain embodiments of the invention. Initially, BMonitor 250
5 monitors the inputs it receives (block 290). These inputs can be from a variety of
6 different sources, such as another node 248, a client computer via network
7 connection 216 (Fig. 3), client operations management console 240, application
8 operations management console 242, an engine 252, a management device 110
9 (Fig. 1), etc.

10 If the received request is a control request (e.g., from one of consoles 240
11 or 242 of Fig. 1, or a management device(s) 110 of Fig. 1), then a check is made
12 (based on the top-level ownership domain) as to whether the requesting device has
13 the necessary rights for the request (block 292). If the requesting device does not
14 have the necessary rights, then BMonitor 250 returns to monitoring inputs (block
15 290) without implementing the request. However, if the requesting device has the
16 necessary rights, then the request is implemented (block 294), and BMonitor 250
17 continues to monitor the inputs it receives (block 290). However, if the received
18 request is a data request (e.g., inbound from another node 248 or a client computer
19 via network connection 216, outbound from an engine 252, etc.), then BMonitor
20 250 either accepts or rejects the request (act 296), and continues to monitor the
21 inputs it receives (block 290). Whether BMonitor 250 accepts a request is
22 dependent on the filters 258 (Fig. 5), as discussed above.

23 Fig. 8 is a flowchart illustrating an exemplary process for handling
24 outbound data requests in accordance with certain embodiments of the invention.
25 The process of Fig. 8 is implemented by BMonitor 250 of Fig. 5, and may be

1 performed in software. The process of Fig. 8 is discussed with additional
2 reference to components in Figs. 1, 3 and 5.

3 Initially, the outbound data request is received (act 300). Controller 254
4 compares the request to outbound request restrictions (act 302). This comparison
5 is accomplished by accessing information corresponding to the data (e.g.,
6 information in a header of a packet that includes the data or information inherent
7 in the data, such as the manner (e.g., which of multiple function calls is used) in
8 which the data request was provided to BMonitor 250) to the outbound request
9 restrictions maintained by filters 258. This comparison allows BMonitor 250 to
10 determine whether it is permissible to pass the outbound data request to the target
11 (act 304). For example, if filters 258 indicate which targets data cannot be sent to,
12 then it is permissible to pass the outbound data request to the target only if the
13 target identifier is not identified in filters 258.

14 If it is permissible to pass the outbound request to the target, then BMonitor
15 250 sends the request to the target (act 306). For example, BMonitor 250 can
16 transmit the request to the appropriate target via transport medium 211 (and
17 possibly network connection 216), or via another connection to network 108.
18 However, if it is not permissible to pass the outbound request to the target, then
19 BMonitor 250 rejects the request (act 308). BMonitor 250 may optionally
20 transmit an indication to the source of the request that it was rejected, or
21 alternatively may simply drop the request.

22 Fig. 9 is a flowchart illustrating an exemplary process for handling inbound
23 data requests in accordance with certain embodiments of the invention. The
24 process of Fig. 9 is implemented by BMonitor 250 of Fig. 5, and may be
25

1 performed in software. The process of Fig. 9 is discussed with additional
2 reference to components in Fig. 5.

3 Initially, the inbound data request is received (act 310). Controller 254
4 compares the request to inbound request restrictions (act 312). This comparison is
5 accomplished by accessing information corresponding to the data to the inbound
6 request restrictions maintained by filters 258. This comparison allows BMonitor
7 250 to determine whether it is permissible for any of software engines 252 to
8 receive the data request (act 314). For example, if filters 258 indicate which
9 sources data can be received from, then it is permissible for an engine 252 to
10 receive the data request only if the source of the data is identified in filters 258.

11 If it is permissible to receive the inbound data request, then BMonitor 250
12 forwards the request to the targeted engine(s) 252 (act 316). However, if it is not
13 permissible to receive the inbound data request from the source, then BMonitor
14 250 rejects the request (act 318). BMonitor 250 may optionally transmit an
15 indication to the source of the request that it was rejected, or alternatively may
16 simply drop the request.

17 **Conclusion**

19 Although the description above uses language that is specific to structural
20 features and/or methodological acts, it is to be understood that the invention
21 defined in the appended claims is not limited to the specific features or acts
22 described. Rather, the specific features and acts are disclosed as exemplary forms
23 of implementing the invention.

CLAIMS

1. One or more computer-readable media having stored thereon a computer program that, when executed by one or more processors of a node in a co-location facility, causes the one or more processors to perform acts including:

beginning and terminating execution of components on the node in response to received commands; and

restricting which other nodes in the co-location facility components that are executing on the node can receive data from and send data to.

2. One or more computer-readable media as recited in claim 1, wherein a plurality of management devices share management responsibility for the node, and wherein beginning and terminating execution of components on the node is restricted to only one of the plurality of management devices at a time.

3. One or more computer-readable media as recited in claim 1, wherein the restricting comprises:

checking whether it is permissible to forward received data to its intended target; and

forwarding the received data to its intended target only if it is permissible to do so.

4. One or more computer-readable media as recited in claim 3, wherein the intended target comprises another node in the co-location facility.

1 5. One or more computer-readable media as recited in claim 3, wherein
2 the intended target comprises at least one of the components executing on the
3 node.

4
5 6. One or more computer-readable media as recited in claim 1, wherein
6 the beginning and terminating execution of components comprises beginning and
7 termination execution of the components based on commands received from an
8 operations console at a location remote from the co-location facility.

9
10 7. One or more computer-readable media as recited in claim 1, wherein
11 one of the components comprises an operating system.

12
13 8. A system comprising:
14 a plurality of node clusters, each node cluster including a plurality of nodes;
15 and
16 wherein each individual node includes a controller to enforce restrictions on
17 which other nodes the individual node can receive data from and which other
18 nodes the individual node can send data to.

19
20 9. A system as recited in claim 8, wherein each individual node further
21 includes a plurality of filters that identify the restrictions.

1 **10.** A system as recited in claim 8, wherein the restrictions prevent the
2 individual node from sending data to or receiving data from another node that is
3 not in the same node cluster as the individual node.

4
5 **11.** A system as recited in claim 8, wherein each individual node
6 includes a network interface adapter that includes the controller.

7
8 **12.** A system as recited in claim 8, wherein for each of the plurality of
9 nodes:

10 a plurality of management devices share management responsibility for the
11 node; and

12 one of the plurality of management devices is given an extended set of
13 management rights over the node, and the remaining management devices is given
14 a more restricted set of management rights over the node.

15
16 **13.** A system as recited in claim 8, wherein the controller in each node
17 is further to terminate and initiate execution of applications on the node in
18 response to requests from an external management device.

19
20 **14.** A system as recited in claim 8, wherein the plurality of node clusters
21 are included in a co-location facility.

22
23 **15.** A method comprising:
24 receiving, at a node in a co-location facility, a first request from a first
25 control console that is local to the co-location facility;

1 implementing the first request;
2 receiving, at the node, a second request from a second control console that
3 is remote from the co-location facility; and
4 implementing the second request.
5

6 **16.** A method as recited in claim 15, wherein the first request comprises
7 hardware operation oriented commands.
8

9 **17.** A method as recited in claim 15, wherein the second request
10 comprises software application control oriented commands.
11

12 **18.** A method as recited in claim 15, wherein the first request
13 corresponds to one of a first set of rights that are granted to the first control
14 console, wherein the second request corresponds to one of a second set of rights
15 that are granted to the second control console, and wherein the first set of rights is
16 more restricted than the second set of rights.
17

18 **19.** One or more computer-readable memories containing a computer
19 program that is executable by a processor to perform the method recited in claim
20 15.
21
22
23
24
25

1 **20.** One or more computer-readable media having stored thereon a
2 computer program that, when executed by one or more processors of a node in a
3 facility, causes the one or more processors to perform acts including:

4 establishing a boundary of a server cluster in the facility, wherein the server
5 cluster includes the node; and

6 altering the boundary of the server cluster based on commands received
7 from a console outside the server cluster.

8
9 **21.** One or more computer-readable media as recited in claim 20,
10 wherein the establishing comprises including a filter that restricts access to another
11 node that is in the facility but that is not in the server cluster.

12
13 **22.** One or more computer-readable media as recited in claim 20,
14 wherein the establishing comprises generating a plurality of filters identifying only
15 other nodes in the server cluster as being permissible to access.

16
17 **23.** One or more computer-readable media as recited in claim 20,
18 wherein the computer program, when executed, further causes the one or more
19 processors to perform acts including executing a software engine in response to a
20 command received from the console.

1 **24.** One or more computer-readable media as recited in claim 20,
2 wherein the computer program, when executed, further causes the one or more
3 processors to perform acts including terminating execution of a software engine in
4 response to a command received from the console.

5
6 **25.** One or more computer-readable media as recited in claim 20,
7 wherein the facility comprises a co-location facility.

8
9 **26.** A system comprising:
10 an interface allowing management devices corresponding to a plurality of
11 management agents responsible for managing the system to access the system; and
12 a controller to operate as a trusted third party mediating interaction among
13 the plurality of management agents by assigning each of the plurality of
14 management agents to a different one of a plurality of ownership domains and
15 restricting the rights of each ownership domain in the system.

16
17 **27.** A system as recited in claim 26, wherein the controller is further to
18 terminate execution of a software engine in the system in response to a request
19 from a management device corresponding to one of the plurality of management
20 agents.

21
22 **28.** A system as recited in claim 26, wherein the controller is further to
23 initiate execution of a software engine in the system in response to a request from
24 a management device corresponding to one of the plurality of management agents.
25

1 **29.** A system as recited in claim 26, wherein one of the plurality of
2 ownership domains is a top-level ownership domain having a first set of rights,
3 and wherein each of the other ownership domains in the plurality of ownership
4 domains has a second set of rights.

5
6 **30.** A system as recited in claim 29, wherein the second set of rights is
7 more restrictive than the first set of rights.

8
9 **31.** A system as recited in claim 29, wherein the first set of rights
10 includes: the right to create new ownership domains, the right to access system
11 memory, the right to access a mass storage device of the system, the right to
12 modify filters in the system, the right to start execution of software engines in the
13 system, the right to stop execution of software engines in the system, the right to
14 debug software engines in the system, the right to change authentication
15 credentials for the ownership domain, the right to modify a storage key for the
16 ownership domain, and the right to subscribe to events engine events, machine
17 events, and packet filter events at the system.

18
19 **32.** A system as recited in claim 29, wherein the second set of rights
20 includes: the right to revoke an existing ownership domain, the right to modify
21 filters in the system, the right to change authentication credentials for the
22 ownership domain, and the right to subscribe to machine events and packet filter
23 events at the system.
24
25

1 **33.** A system as recited in claim 29, wherein the first set of rights
2 includes: the right to create new ownership domains, the right to access system
3 memory, the right to access a mass storage device of the system, and the right to
4 modify filters in the system.

5
6 **34.** A system as recited in claim 29, wherein the second set of rights
7 includes: the right to revoke an existing ownership domain and the right to modify
8 filters in the system, including the right to add a filter that cannot be subverted by
9 a management agent assigned to the top-level ownership domain.

10
11 **35.** A system as recited in claim 29, wherein the controller allows a
12 device corresponding to any one of the other ownership domains to revoke the
13 top-level ownership domain, and wherein the controller erases a system memory
14 during the revocation process.

15
16 **36.** A system as recited in claim 26, wherein only one of the plurality of
17 management agents can correspond to a top-level ownership domain at a time, and
18 wherein any of the other management agents can revoke the top-level ownership
19 domain.

20
21
22
23
24
25

1 **37.** A system as recited in claim 26, wherein only one of the plurality of
2 management agents can correspond to a top-level ownership domain at a time, and
3 wherein the one management agent can create a new ownership domain for a new
4 management agent, and wherein the new ownership domain becomes the new top-
5 level ownership domain.

6
7 **38.** A system as recited in claim 26, wherein only one of the plurality of
8 management agents can correspond to a top-level ownership domain at a time,
9 wherein which of the plurality of management agents corresponds to the top-level
10 ownership domain at any given time can vary over time, and wherein the
11 controller erases a system memory each time a change occurs in which of the
12 plurality of management agents corresponds to the top-level ownership domain.

13
14 **39.** A system as recited in claim 26, wherein the system comprises a
15 node in a co-location facility.

16
17 **40.** A method comprising:
18 associating each of a plurality of management agents with one of a plurality
19 of ownership domains, wherein each of the plurality of management agents is
20 responsible for managing at least a portion of a computer and is external to the
21 computer;

22 allowing only one of the plurality of management agents to have an
23 extended set of rights to the computer at a time, and assigning the remaining
24 management devices a more limited set of rights; and
25

1 restricting which requests from management devices corresponding to the
2 plurality of management agents are carried out based at least in part on the rights
3 of the management agent.

4
5 **41.** A method as recited in claim 40, where each of the plurality of
6 management agents corresponds to one or more management devices that are
7 coupled to the computer.

8
9 **42.** A method as recited in claim 40, wherein the extended set of rights
10 includes: the right to create new ownership domains, the right to access system
11 memory, the right to access a mass storage device of the system, the right to
12 modify filters in the system, the right to start execution of software engines in the
13 system, the right to stop execution of software engines in the system, the right to
14 debug software engines in the system, the right to change authentication
15 credentials for the ownership domain, the right to modify a storage key for the
16 ownership domain, and the right to subscribe to events engine events, machine
17 events, and packet filter events at the system.

18
19 **43.** A method as recited in claim 40, wherein the more limited set of
20 rights includes: the right to revoke an existing ownership domain, the right to
21 modify filters in the system, the right to change authentication credentials for the
22 ownership domain, and the right to subscribe to machine events and packet filter
23 events at the system.

1
2 **49.** A method as recited in claim 40, further comprising terminating
3 execution of a software engine in the computer in response to a request from a
4 management device corresponding the one management agent having the extended
5 set of rights.

6
7 **50.** A method as recited in claim 40, further comprising initiating
8 execution of a software engine in the computer in response to a request from a
9 management device corresponding the one management agent having the extended
10 set of rights.

11
12 **51.** A method as recited in claim 40, wherein the computer comprises a
13 node in a co-location facility.

14
15 **52.** One or more computer-readable memories containing a computer
16 program that is executable by a processor to perform the method recited in claim
17 40.

1 **ABSTRACT**

2 A controller, referred to as the "BMonitor", is situated on a computer. The
3 BMonitor includes a plurality of filters that identify where data can be sent to
4 and/or received from, such as another node in a co-location facility or a client
5 computer coupled to the computer via the Internet. The BMonitor further receives
6 and implements requests from external sources regarding the management of
7 software components executing on the computer, allowing such external sources to
8 initiate, terminate, debug, etc. software components on the computer.
9 Additionally, the BMonitor operates as a trusted third party mediating interaction
10 among multiple external sources managing the computer.
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

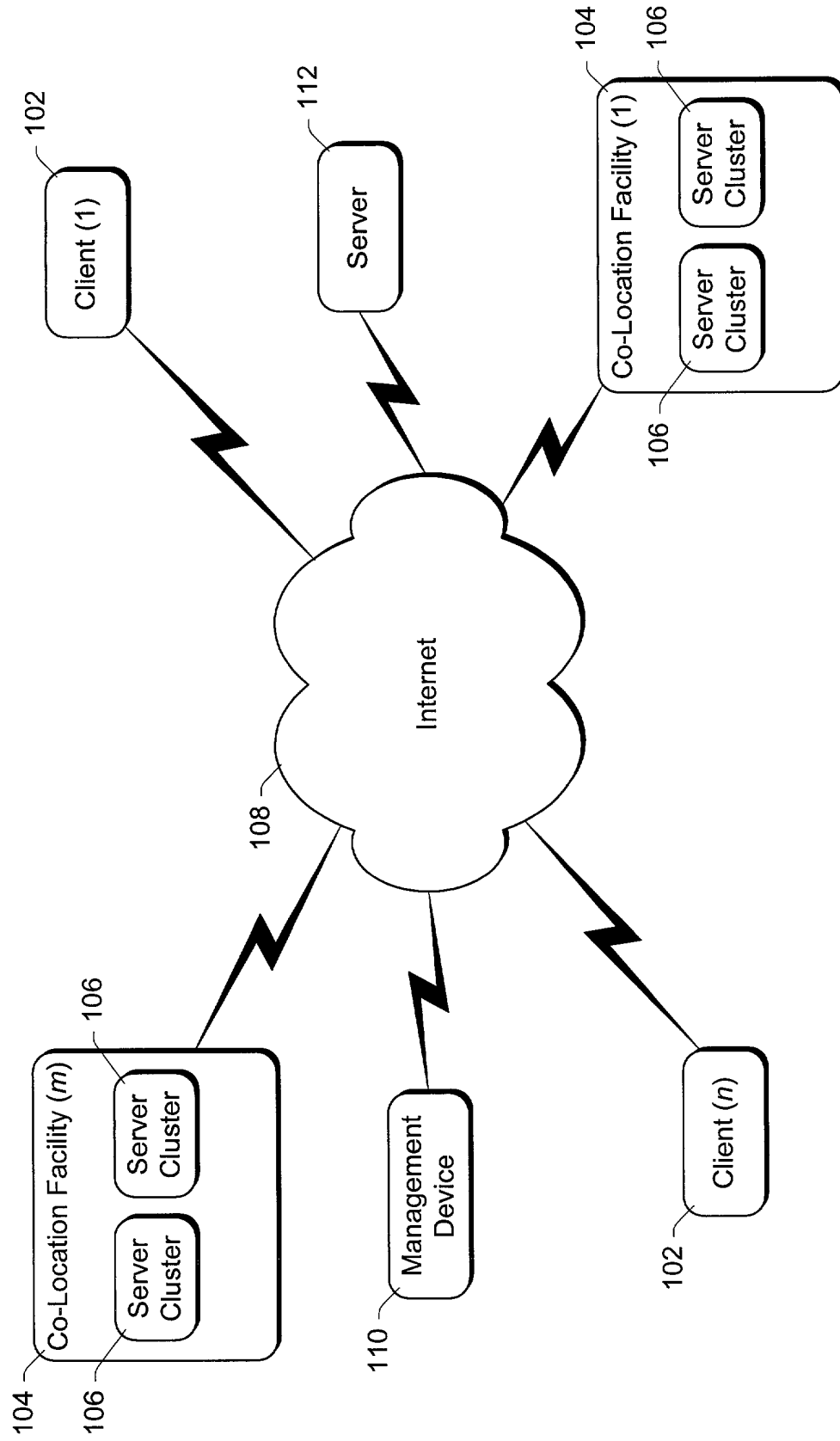
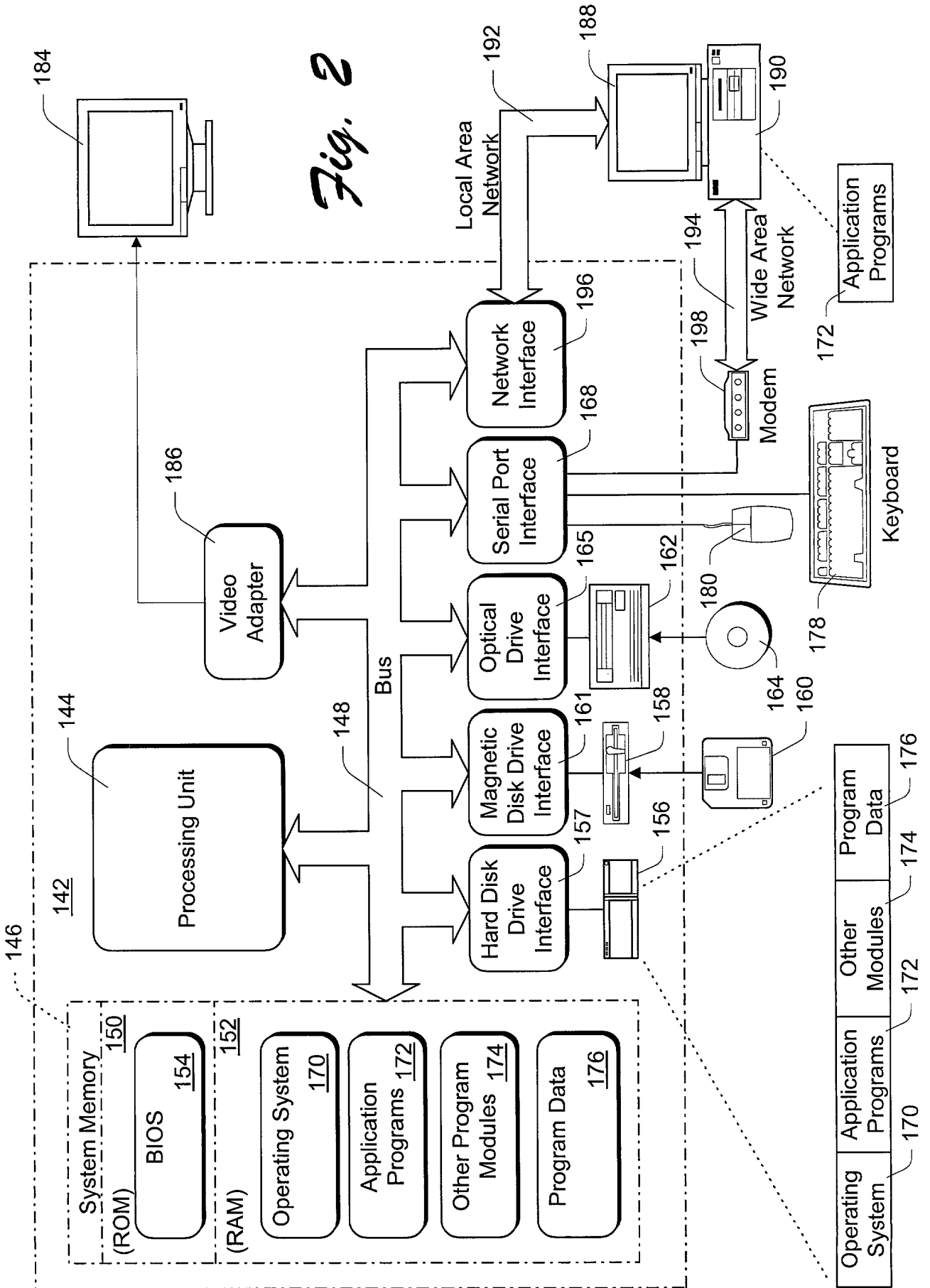


Fig. 1

Fig. 2



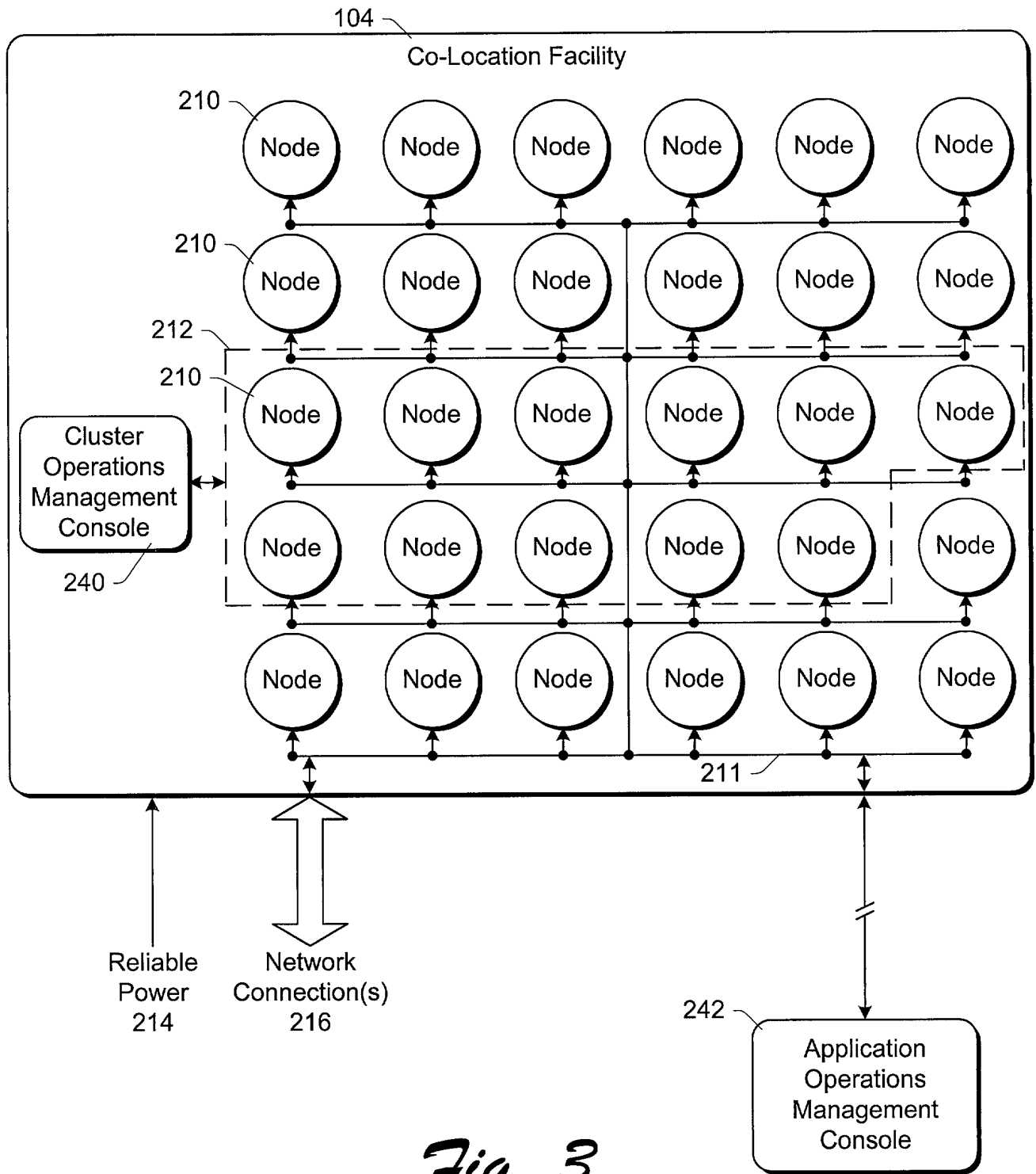


Fig. 3

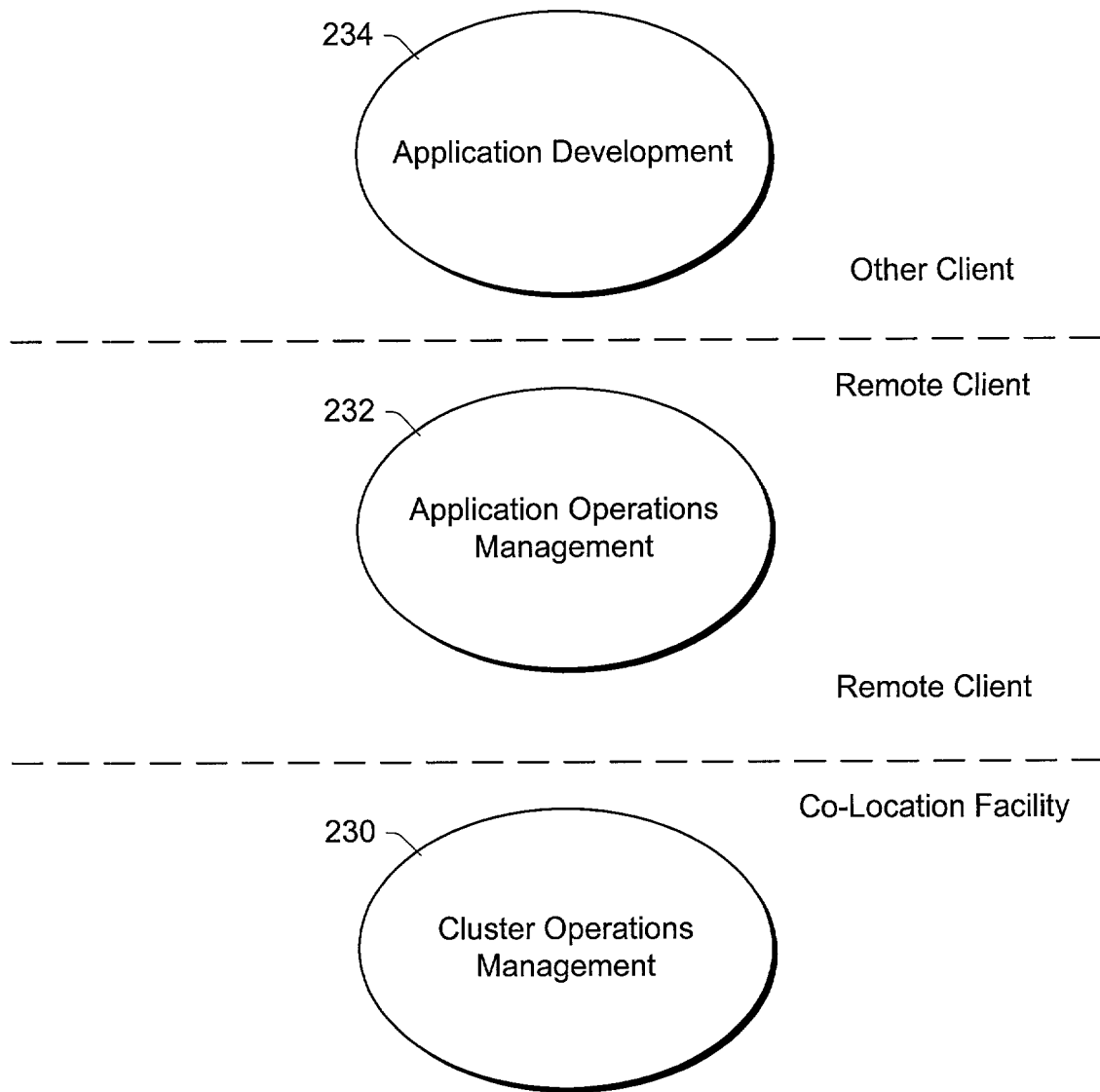


Fig. 4

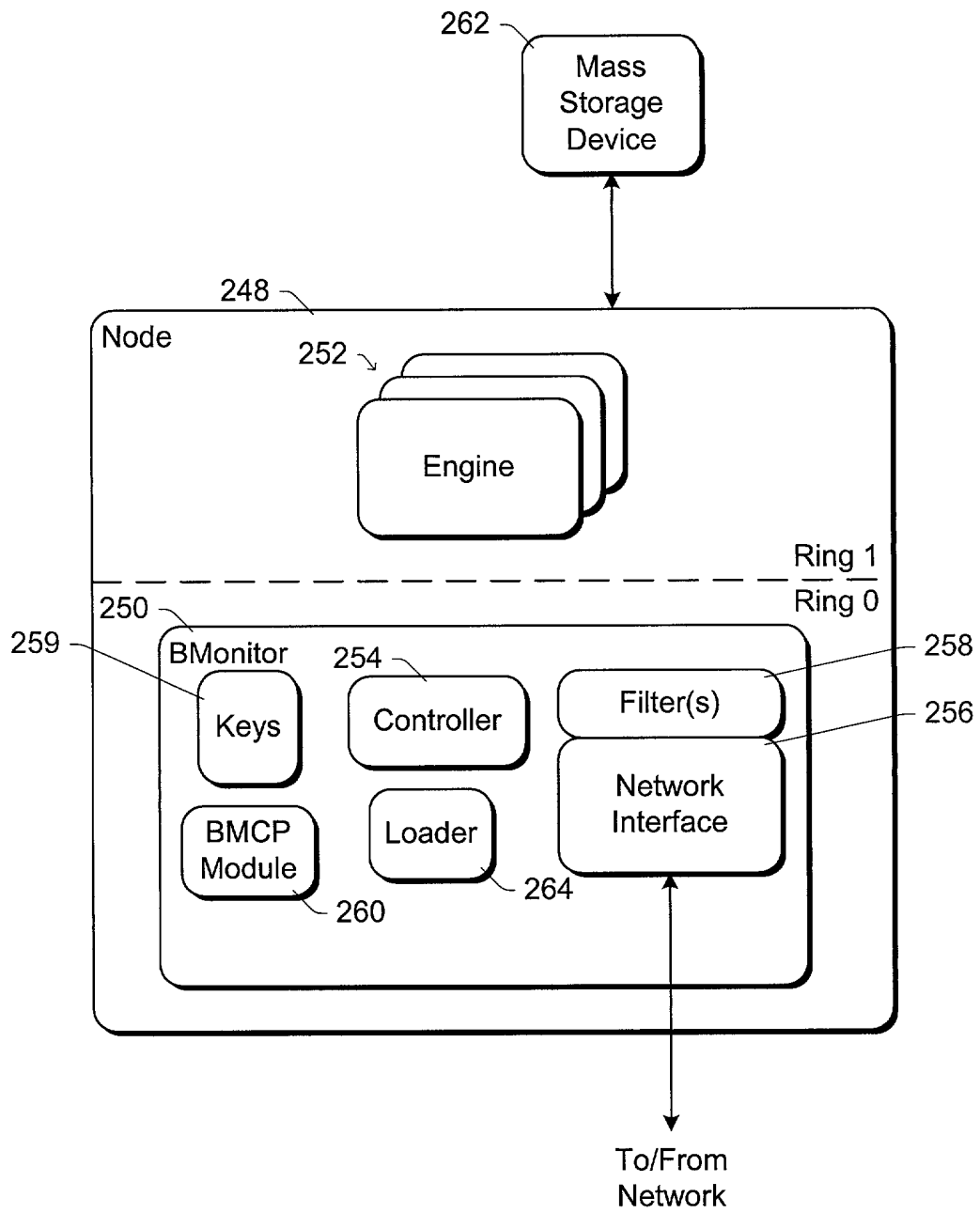
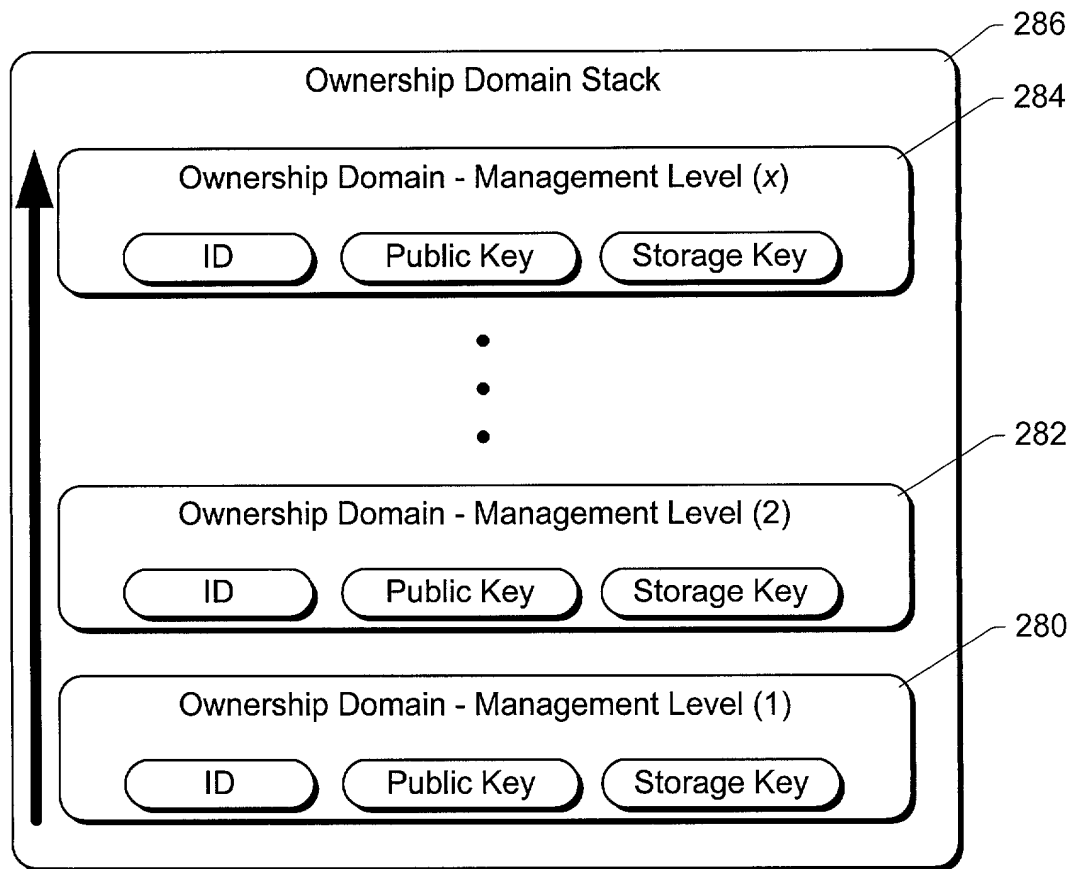


Fig. 5

*Fig. 6*

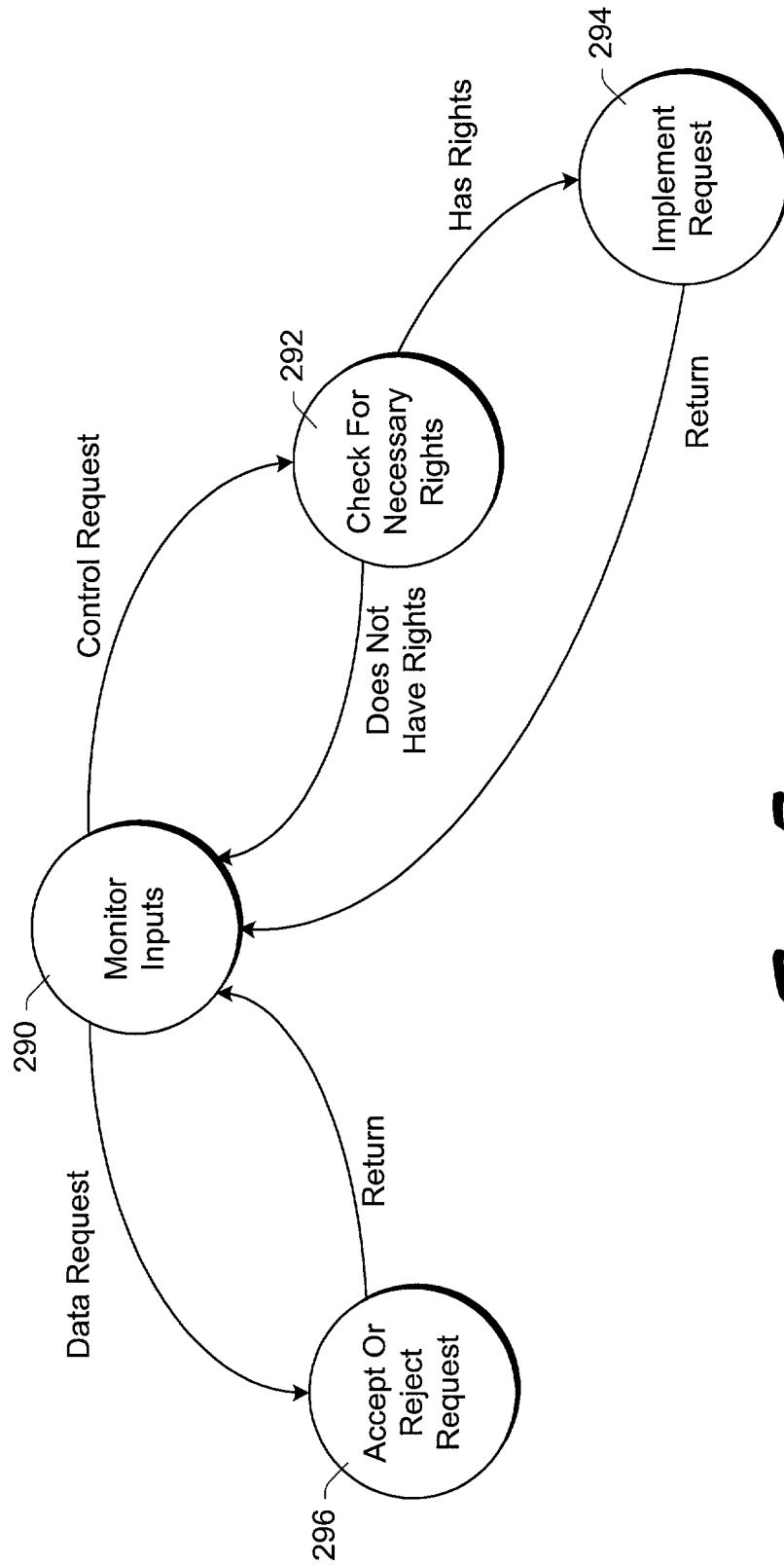


Fig. 7

000007-00000000

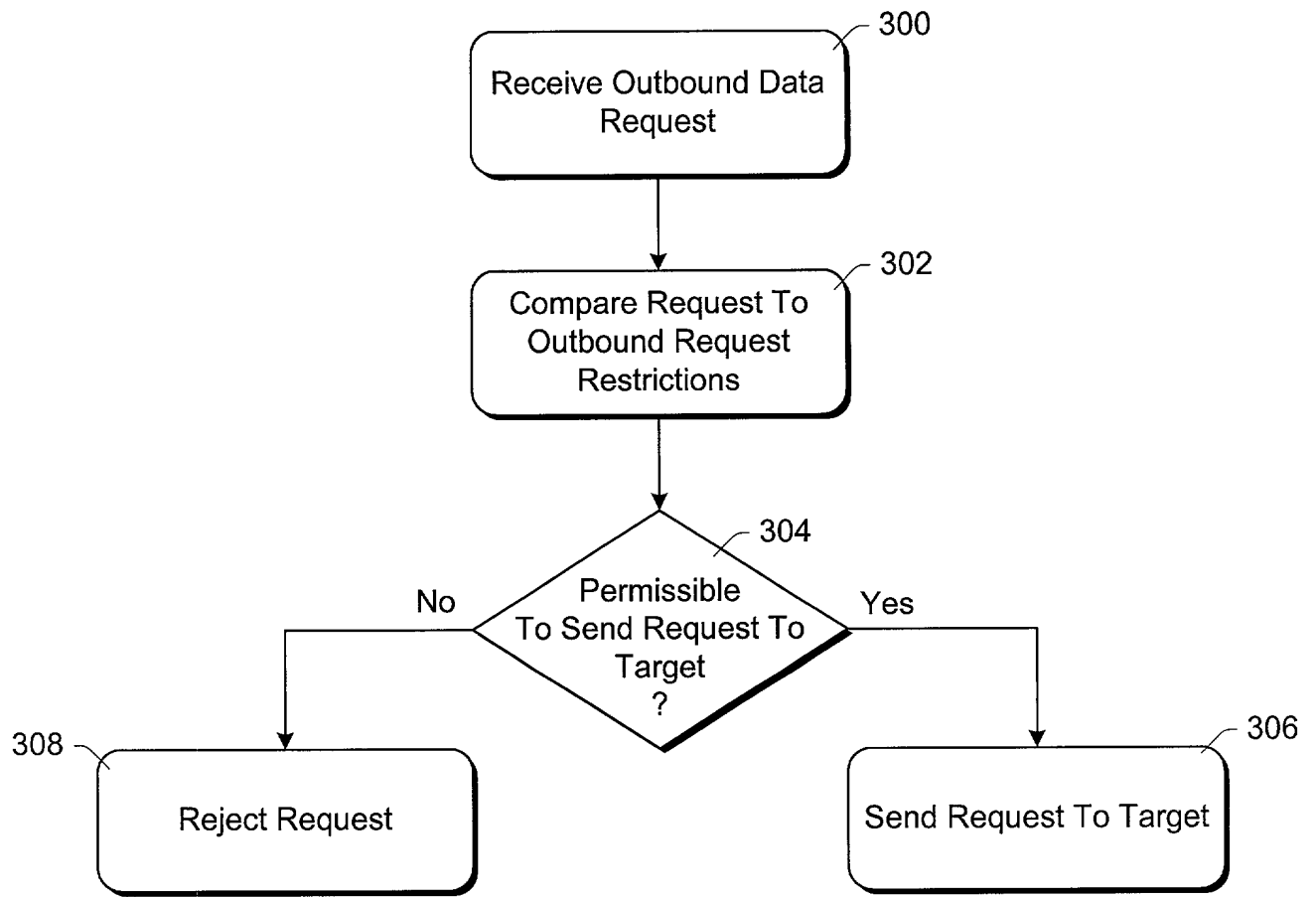


Fig. 8

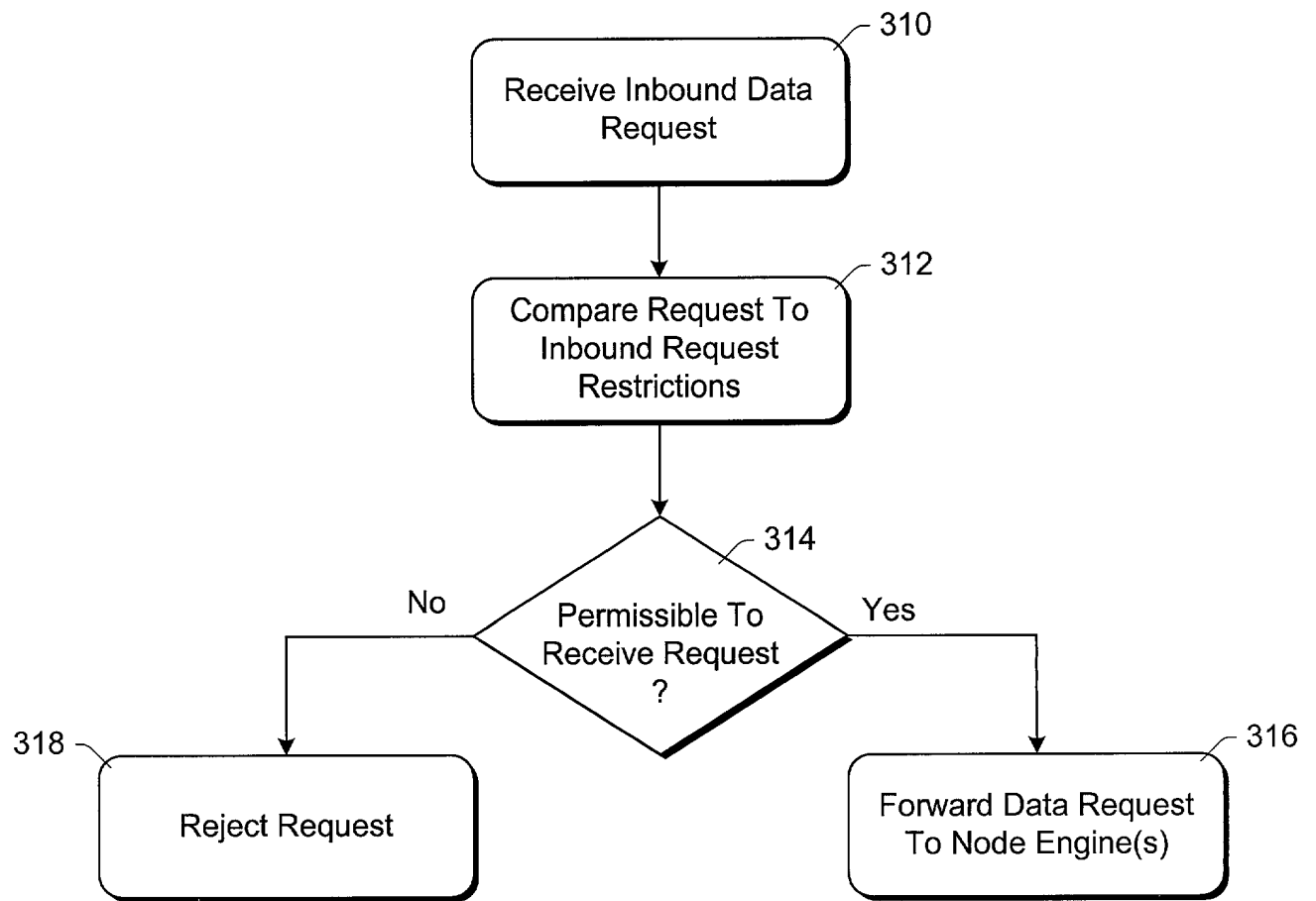


Fig. 9